Naval Research Laboratory
Washington, DC 20375-5320

NRL/MR/5540--09-9220

An Infrastructure for Multi-Level Secure Service-Oriented Architecture (MLS-SOA) Using the Multiple Single-Level Approach

JIM LUO AND MYONG KANG
Center for High Assurance Computer Systems
Information Technology Division

December 17, 2009

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
17-12-2009	Memorandum Report	September 2008 - September 2009

4. TITLE AND SUBTITLE

An Infrastructure for Multi-Level Secure Service-Oriented Architecture (MLS-SOA) Using the Multiple Single-Level Approach

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
0602435N

6. AUTHOR(S)

Jim Luo and Myong Kang

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
4555 Overlook Avenue, SW
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/5540--09-9220

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR / MONITOR'S ACRONYM(S)

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

SOA is the premier framework for integrating complex heterogeneous computing systems in business and government. To utilize SOA in sensitive military systems, however, the requirements for multi-level security (MLS) must be addressed. This paper presents a framework for adding MLS capabilities to existing SOA infrastructure. Specifically, it will allow clients in High to securely utilize services in Low. MLS issues including covert channels, release of identity information, and inference attacks are addressed. Our scheme uses the multiple single-levels (MSL) approach to offer a practical solution that leverages existing technology. It can be deployed immediately without developing and certifying new high assurance MLS components. The MLS-SOA infrastructure can be installed cleanly on top of regular SOA and MLS components. Operations of existing infrastructure, services, and applications will not be affected. Capabilities provided by this scheme can also be applied to other non-MLS settings with similar requirements for separation and anonymity.

15. SUBJECT TERMS
MLS SOA
MSL

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Jim Luo
Unclassified	Unclassified	Unclassified	UL	18	19b. TELEPHONE NUMBER *(include area code)* (202) 767-3381

An Infrastructure for Multi-Level Secure Service-Oriented Architecture (MLS-SOA) Using the Multiple Single-Level Approach

1. Introduction

Service Oriented Architecture (SOA) is a style of software development and integration characterized by loose coupling among interacting software agents. It follows the *publish-discover-invoke* paradigm to create *ad hoc* applications from a collection of smaller service modules. This approach allows for rapid and flexible application composition and deployment while improving the overall system quality. It is the premier integration and architecture framework in today's complex and heterogeneous computing environments.

Figure 1: SOA paradigm.

The US military is showing a great deal of interest in developing SOA technology for all of its information technology needs including at the tactical level [1]. The mission critical nature of military information technology creates a whole new set of requirements for SOA that are not fully addressed by solutions currently available for the private sector. A requirement that is completely unique to the military setting is multi-level security (MLS) [2-4].

In an MLS environment, clients in a lower classification level (Low) should not access services in a high classification level (High). However, the reverse is not necessarily true. There are many situations where it is highly desirable or necessary for clients in High to directly access services hosted in Low.

Typically, many more services are available in the lower security domain. Non-critical services such as weather forecasting, mapping, and logistics usually reside in unclassified domains, but are needed by clients in classified domains. Information systems outside of the military in both the public and commercial sectors provide services that are potentially useful for all phase of military operations at all levels of classification. Sensitive military information systems in the classified domain often acquire data from civilian sources and need to directly interface with those systems. Access to operational level systems in the classified domains in turn may be needed by intelligence or planning clients in top-secret domains.

Existing MLS security models for domain separation cannot be applied to the SOA environment. For example, the request-response interaction that lies at the heart of SOA inherently violates the Bell-LaPadula model [2] if it crosses domain boundaries. Current MLS models forbid SOA interactions across security boundaries. Therefore, it is only possible to deploy SOA in a single security level through duplication of the required services. Duplication is obviously burdensome and inefficient. Furthermore, it may be impossible to elevate certain services to High due to conflicting physical, organizational or

Manuscript approved September 10, 2009.

technical requirements. The MLS limitations for SOA must be adequately addressed before it can be widely deployed in military settings.

MLS requirements call for strict separation between hierarchical security levels. Higher security levels are more restrictive. That means activities in High should not be visible to Low, but High is not prohibited from observing activities in Low. That means, in the SOA context, clients in High may be able to utilize services in Low without compromising core MLS principles as long as certain additional security conditions are met.

This paper explores those security conditions and presents an MLS enabled security architecture for SOA. The resulting system design and infrastructure will be referred to as MLS-SOA. The design is based on the multiple single-levels (MSL) [5, 6] implementation of MLS. This approach will not produce a trusted computing platform (TCP). Instead, it offers a practical solution for adding MLS capabilities to existing technologies using existing high-assurance components. MLS-SOA will leverage the standard MLS infrastructure already in place to enable core functionalities of SOA across security levels.

The rest of this paper is organized as follows. Section 2 presents an overview of the SOA stack. Section 3 provides background information on MLS and MSL, and explores the security issues specific to SOA. Section 4 specifies the functional and security requirements for MLS-SOA. Section 5 describes the design of the security architecture. Section 6 examines related work in the area of MLS. Implications of MLS-SOA are discussed in Section 7 and the related work is discussed in section 8. Section 9 offers a conclusion.

2. SOA Overview

Most web services restrict access to authorized clients. Therefore, the SOA paradigm is better summarized as *publish-discover-authenticate-invoke* (Figure 2). Services usually do not perform identity management and authentication themselves. Instead, they rely on a centralized identity provider for the entire domain. Clients authenticate to the identity provider, which is in turn trusted by the services.

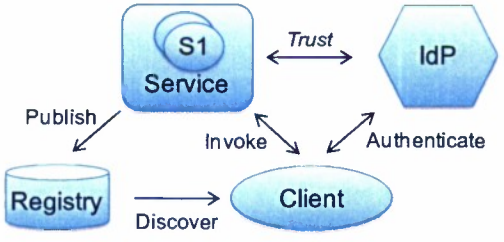

Figure 2: SOA paradigm with Authentication.

In a federated setting, trust is maintained between the services and the identity providers in partner domains. Clients that authenticate with their own identity provider can access services in different domains as long as its identity provider is trusted by the services (Figure 3). Service discovery across domains can be performed by directly querying partner repositories. However, repository replication is more efficient. Service descriptions are transferred to the local repository through a standard interface and queries can then be performed locally.

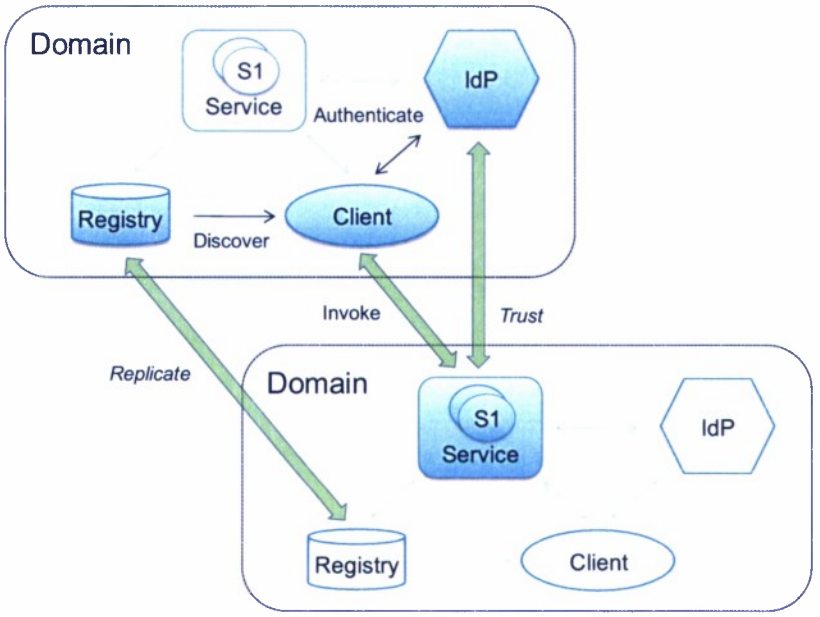

Figure 3: Cross-domain service invocation.

There are numerous commercial and open source technologies for SOA. We choose a stack based on accepted and open standards. Universal Description Discovery and Integration (UDDI) will serve as the platform for publishing and discovery [7]. Security Assertion Markup Language (SAML) will serve as the basis for communicating authentication information in a federated environment [8]. Service invocation will be supported at the HTTP/HTTPS level. That means the resulting infrastructure will operate with SOAP [9] based as well as browser based web services.

3. MLS in the SOA Computing Environment

3.1. MLS Background

Full MLS systems are rarely deployed. Trusted computing platforms are extremely difficult and costly to develop and certify. Most military information systems are deployed using multiple single-level (MSL) implementations [5, 6]. Computers and networks in different security levels are strongly separated, for example, by using completely separate sets of hardware. Inside each security level, components do not require high assurance. Standard commodity hardware, OS and software can be utilized without MLS implications. High assurance devices are only needed to mediate communications that cross security levels. Cross-domain solutions (CDS) enforce releasability policies and often serve to control covert channels [10, 11]. They are widely deployed in military networks currently and are an integral part of the MLS infrastructure. Most CDS follow the Bell-LaPadula model with downgrading capability to allow two-way communications. They are also capable of processing XML data [10]. These devices can be used directly to mediate the two-way XML based traffic in SOA.

The MSL approach translates well into the SOA context. Current SOA systems that are deployed inside single security levels can be connected together using CDS. Advantages of this approach are 1) it leverages existing technology and 2) new high assurance components do not have to be developed and accredited. However, the SOA paradigm creates several new MLS related challenges that must be specifically addressed by the security architecture.

3.2. SOA Specific Issues

3.2.1. Covert Channels

SOA is interactive in nature. The increase in communications across security boundaries creates a bigger potential for covert channels. There are at least three potential covert channels during service invocation. The first is a timing channel using the frequency at which services in Low are invoked. Timing of multiple service invocations can be modulated to send messages. The second related covert channel is a storage channel using the order in which services are invoked. The specific services can serve as an alphabet for messages. Neither of these types of covert channels can be completely eliminated, but they can be controlled by limiting their potential bandwidth. They are well studied and existing CDS technology already address them. Specific methods employed by the CDS include modulating and limiting the rate of service invocations and rearranging their order [10-12].

The third covert channel involves hiding data in otherwise releasable service invocation parameters, for example, by using stenography. Passing of parameters from High to Low is an overt channel for data flow. Additional data hidden in the parameters constitutes the covert channel. This is by far the biggest potential covert channel. Releasability of information is in theory enforced by the down-grader in the CDS infrastructure. However, capability of existing down-graders is relatively limited. Standard filters using techniques such as dirty word checking will not be able to detect more sophisticated methods of hiding information. Discussion of covert channels in MLS-SOA must take into account the large overt channel that must exist.

The decision to allow service invocation from High to Low necessitates information flow across domain boundaries. A tradeoff between security and functionality is inherent in that decision. Covert channels in SOA environments are indeed similar in nature to traditional MLS distributed computing environments. However, the highly interactive nature of SOA and its potential to generate large amounts traffic to a large number of different domains multiplies the problem. The MLS-SOA infrastructure does not need to directly address covert channels because they are already addressed by existing CDS infrastructure. However, cross-domain interactions in MLS-SOA need to be minimized. This would make covert channels in MLS-SOA no worse than existing cross-domain applications. It would also simplify the additional policy specification and configuration of the CDS necessary to support MLS-SOA.

3.2.2. Release of Identity Information

SOA is loosely coupled. Clients are able to interact with services in many different domains. Authentication to those services necessitates the release of identity information. This is not an issue in traditional MLS systems that are tightly coupled. The static nature of that coupling means components are implicitly aware of the identity of their counterparts and the source data. SOA systems are much more dynamic. Services may receive requests from clients in many different organizations. If service invocation is enabled from High to Low, revealing client identity information constitutes an information release from High to Low. This applies to not only identity of individual clients, but also the identity of their domains.

Authentication is a required facet of the SOA paradigm in most practical situations. Traditional models tie trust directly to identity. A new method of establishing trust is needed where anonymity is preserved. Service providers must be able to verify that clients are authorized to access the service without directly knowing who the clients are.

3.2.3. Prevention of Inference

The inference and aggregation problem in the context of MLS has been studied a great deal in the past [13, 14], however, satisfactory solutions or security models have yet to be developed. The interactive and loosely coupled nature of SOA brings this classic problem again to the forefront. Multiple service invocations, combined with the identity of the client, could allow adversaries in Low to infer non-releasable information in High. No classified information is released in any of the individual service invocation. However, when aggregated, the services requested, the service parameters, and the client

identity can paint a picture that is not releasable. For example, a weather forecast request for a specific time in a region combined with mapping service request for a specific building provides the exact time and place for a potential action by a specific organization. Spreading out service request to multiple service providers reduces the likelihood that a single entity can compile all the data. However, lower security domains as a whole cannot be trusted by High.

Inference and aggregation depends on the ability to tie multiple service requests to a single actor. The prevention of inference issue is related to the release of identity information issue but goes further. Entities in Low should not be able to determine that multiple service requests came from the same source.

4. MLS-SOA Requirements

4.1. Functional Requirements

The core functionalities of the standard single-level SOA *publish-discover-authenticate-invoke* paradigm must be retained in the MLS-SOA security architecture. The functional requirements for MLS-SOA can be summarized as follows:

1. Cross-domain service publication and discovery.
2. Cross-domain authentication and authorization.
3. Cross-domain service invocation.

The MLS-SOA security architecture must interoperate fully with the existing SOA infrastructure. Interaction with the existing components should only go through the standard interfaces they already support, in this case, UDDI, SAML, and HTTP. The MLS-SOA components should not interfere with the operation of existing single-level SOA services, infrastructure and applications. Many single-level SOA systems deployed in the military are mission critical. Owners would be more willing to deploy new capabilities if they do not pose a risk their normal operations or negate their previous investments in technology.

4.2. MLS Requirements

The MLS requirements for MLS-SOA can be summarized as follows:

1. Minimize the communications and interactions between High and Low that cross the classification domain boundary.
2. Hide the identity of High service client from Low service providers. Not only should the identity of the specific client be hidden, but also the identity of the client domain should also be hidden.
3. Mitigate inference and aggregation attack from the Low domain. Services in the Low domain must not be able to correlate multiple service invocations to the same source.

These requirements will be applied to an MSL implementation. Security domains will be connected through one or more cross-domain solutions. We assume the CDS will properly mediate data flow to enforce releasability and control covert channels according to the MLS policy. The CDS will block both accidental and malicious release of sensitive data. The specifics of the CDS are outside the scope of this paper. Existing CDS devices capable of processing XML traffic can be used. The CDS will be the only high assurance component in the security architecture. Satisfaction of the MLS requirements will not rely on the integrity or cooperation of any components in the Low domain.

5. System Design

5.1. Client, Service, and Identity Provider Proxies

The diagram in Figure 4 summarizes the normal SAML protocol for service invocation [8]. The client and the service provider reside in different federated domains. This diagram shows the browser profile. The enhanced client or proxy (ECP) profile follows a similar traffic pattern.

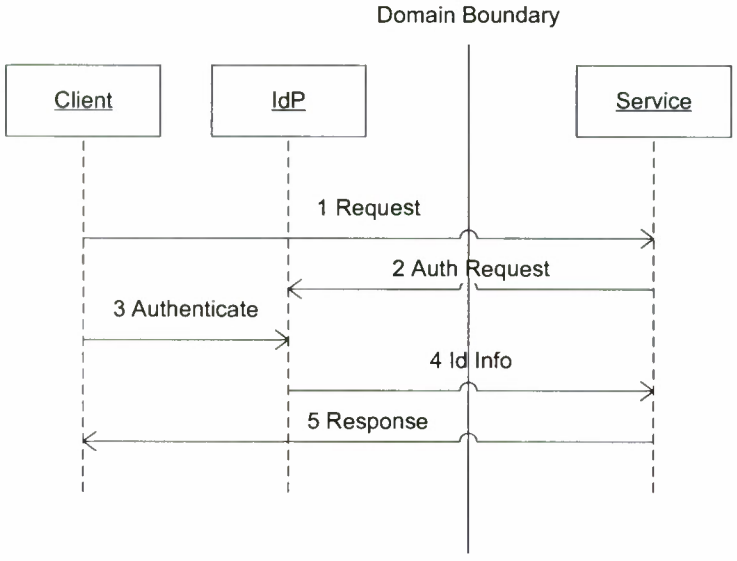

Figure 4: SAML service invocation protocol.

The client connects to the service endpoint (1). The service provider redirects the client to its identity provider with an authentication request (2). The client authenticates with its identity provider (3). After authentication, the client is redirected back to the service provider with its identity information (4). The service provider performs authorization with the identity information and a response is returned to the client (5).

Service invocation should only involve a single round trip: the request going from the client to the service and the response going from the service to the client. The SAML protocol crosses the domain boundary four times (more if request for additional attributes are made). Our approach is to create proxies for SOA components so that SAML redirections only take place within the same security level. Traffic that crosses security levels between the service and client proxies will follow our own custom formats instead of the SAML protocol. The proxies will replicate the respective SAML interfaces to make and receive redirections from existing SOA components. On the client side, a service proxy will play the role of the service provider for the client and identity provider. On the service side, a client proxy and an identity provider proxy will play the respective roles for the service provider. This approach allows us to customize the traffic crossing domain boundaries while maintaining interoperability with existing SOA components that speak SAML.

Figure 5 shows the proxies and their interaction with existing SOA components. The shaded boxes are standard SOA components involved in the service invocation. The double lined boxes are MLS-SOA components added to the existing SOA infrastructure. The small arrows indicate interactions using standard SOA interfaces. The wide arrow indicates customized MLS-SOA interaction across CDS.

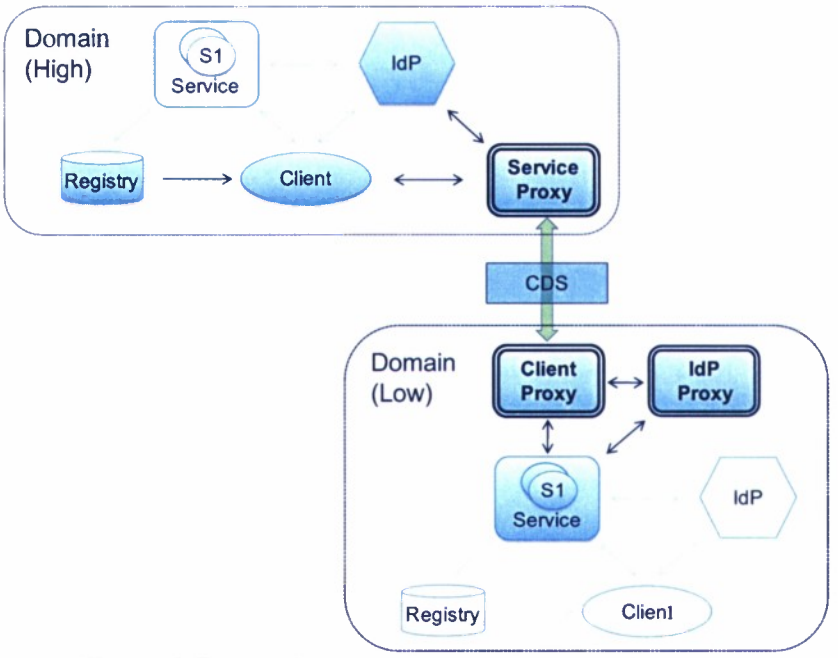

Figure 5: Proxies to eliminate cross-domain redirections.

In the standard SAML protocol, the service provider queries the identity provider for security attributes it needs to perform authorization. In our scheme, the service proxy must proactively gather the required security attributes and bundle them with the service request. That means the security attributes required for authorization must be known in advance. This information is usually agreed upon as part of a memorandum of understanding (MOU) between domains. They can be ascertained while the federation is being created and simply stored in a database in the client domain. The security attributes released to Low domains should not compromise the anonymity of the client. This will be enforced by both the CDS and the MLS-SOA infrastructure (discussed in the next section).

To invoke services in Low, High clients will connect to the service proxy instead of the actual service provider. The service description should be modified to reflect this fact. The connection string in the service description will point to the service proxy with the actual service provider endpoint and client proxy endpoint encoded as parameters.

http://noaa.gov/weather?cmd=get_temp

⇩

http://sproxy?endpoint=base64(http://noaa.gov/weather...)&client_proxy=base64(http://cproxy.noaa.gov)

This change is transparent and client can use the modified connection string normally. The proxies will parse the parameters and make the appropriate connections for the client.

Figure 6 summarizes the MLS-SOA protocol for service invocation. The client creates a service request and sends it to the service proxy (1). The service proxy redirects the client to the identity provider for authentication (2). The client authenticates (3) and gets redirected back to the service proxy with its identity information (4). The service request and the client identity information are combined and sent to the client proxy across the domain boundary (5). The client proxy takes the service request and connects to the service provider (6). The service provider redirects the client to the identity provider proxy with an authentication request (7). The identity provider proxy gets the identity information from the client proxy (8) and redirects the client back to the service provider (9). The service provider authorizes the identity information and grants access. The response is sent back to the client proxy, which sends it back to the service proxy, which in turn sends it back to the High client (10).

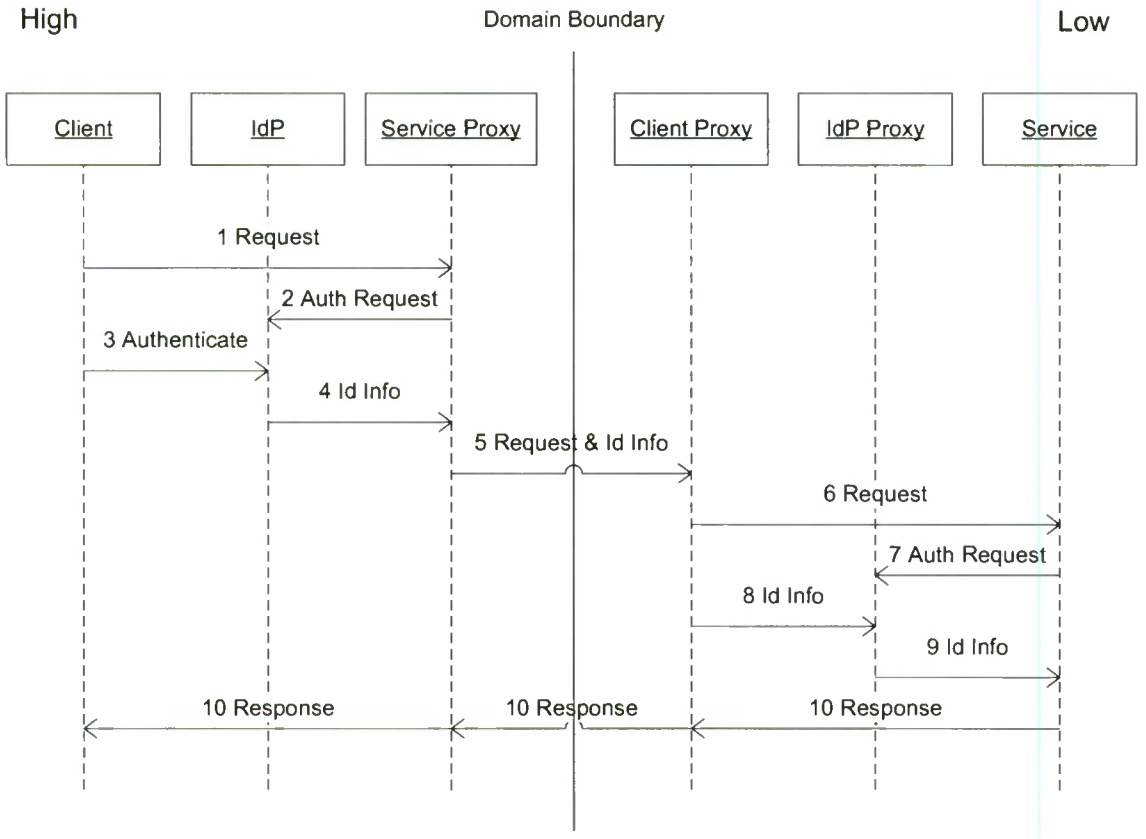

Figure 6: MLS-SOA service invocation.

In the diagram, steps 2 – 4 and 6 – 9 follow the SAML protocol. The communications in step 5 and 10 are between two MLS-SOA components and use a customized format. This is the only traffic that crosses the domain boundary and will be mediated by the CDS.

5.2. Anonymous Authentication

In standard SAML, client identity information from the identity provider is used directly by the service provider to perform authorization. However, the MLS requirement does not allow the release of identity information of the High clients. That means the service provider must authorize without knowing the identity of the client. Furthermore, the domain from which the request originates must also be hidden. The service provider should not be able to determine which domain the client came from.

5.2.1. Client Identity Hiding.

The cross-domain service invocation protocol using proxies was described in the previous section. It showed the flow of identity information from the client identity provider to the service provider. The identity information goes through the service proxy (step 4), the client proxy (step 5), and the identity provider proxy (step 8). In this scheme, identity information is filtered at service proxy and then transformed at the identity provider proxy.

The SOA identity provider is not aware of MLS requirements and will return a large number of security attributes. The service proxy will sanitize those security attributes so that they can be released to Low. This is done according to the underlying MLS policy and the MOU between the domains. Attributes

that are not releasable can be simply stripped or changed to an obfuscated form. It is entirely possible that all attributes are stripped. The MOU could simply state that all users authenticated in High can access resources in Low. Non-releasable identity information will not be sent to any components in Low since sanitation is performed in the High domain. The identity information referred to in step 5 of Figure 6 is the sanitized. The sanitation is performed by the service proxy, which is not a trusted component. However, the CDS, which is a trusted component, mediates all information flow and will ultimately be responsible for preventing the release of non-releasable identity information.

The identity provider proxy will create a new set of security attributes for the service provider to authorize. Specific services may require identifying attributes about their clients such as name, email address, and affiliation. Since those attributes are not released, they have to be created. In essence, a fake identity is generated for the High client to access the service. The values of the fake identity depend on the specific policy of the service provider domain and the individual services. The sanitized client identity information could be used as input. The algorithm for generating the security attributes must preserve client anonymity and prevent inference attacks. For example, it would be ill advised to simply hash the username. Even though the identity is obfuscated, subsequent service invocations can be attributed the same client. Generation of the fake identities is implementation specific and is outside the scope of this paper. The simplest implementation would be to create a single user profile for all High clients.

5.2.2. Domain Identity Hiding

MLS-SOA requirements go further than just hiding the identity information for specific clients. Components in Low should not be able to identify the originating domains of High clients. That means the service provider has to trust the identity provider of the client without knowing who it is. The traditional methods of establishing trust between interacting SOA components using certificates and signatures would not work for MLS-SOA. In order to verify a signature, the identity of the signer must be known. Trust is directly tied to identity.

In the MLS-SOA security architecture, trust is anonymized between the service proxy in High and the client proxy in Low. This is the step in the services invocation protocol that crosses domain boundaries and the only one that requires anonymity. Other trust relationships between the proxies and the SOA components inside a single security level can be established with traditional methods using certificates.

Our approach for establishing anonymous trust between the service proxy and the client proxy rely on anonymous tickets. The process breaks down to two phases. Ticket generation phase happens at an indeterminate time prior to service invocation. The service proxy from the client High domain authenticates itself to the client proxy in the service Low domain using conventional methods and acquires the anonymous tickets. During the service invocation phase, the presentation of a valid anonymous ticket proofs that the service proxy has been authenticated previously. However, the client proxy cannot determine which of the many service proxy it previously authenticated is currently making the service invocation.

The anonymous ticket approach is based on blind signatures [15]. The blind signature algorithm allows users to obtain signatures on messages without revealing the content of the messages to the signer. It is very useful in applications where both authentication and anonymity are required such as digital cash, voting machines, or pervasive computing [16-18].

During ticket generation (Figure 7), the service proxy in the High client domain will generate a cryptographically random alpha-numeric string referred to as a nonce (1), blind it using a secret blinding factor (2), and send its authentication credentials along with the blinded nonce to the client proxy in the Low service domain (3). The client proxy verifies the credentials (4) and signs the blinded nonce to generate a blinded signature (5). The blinded signature is returned (6) and the service proxy un-blinds it using the same secret blinding factor to retrieve the signature on the original nonce (7). The nonce and its signature constitute an anonymous ticket.

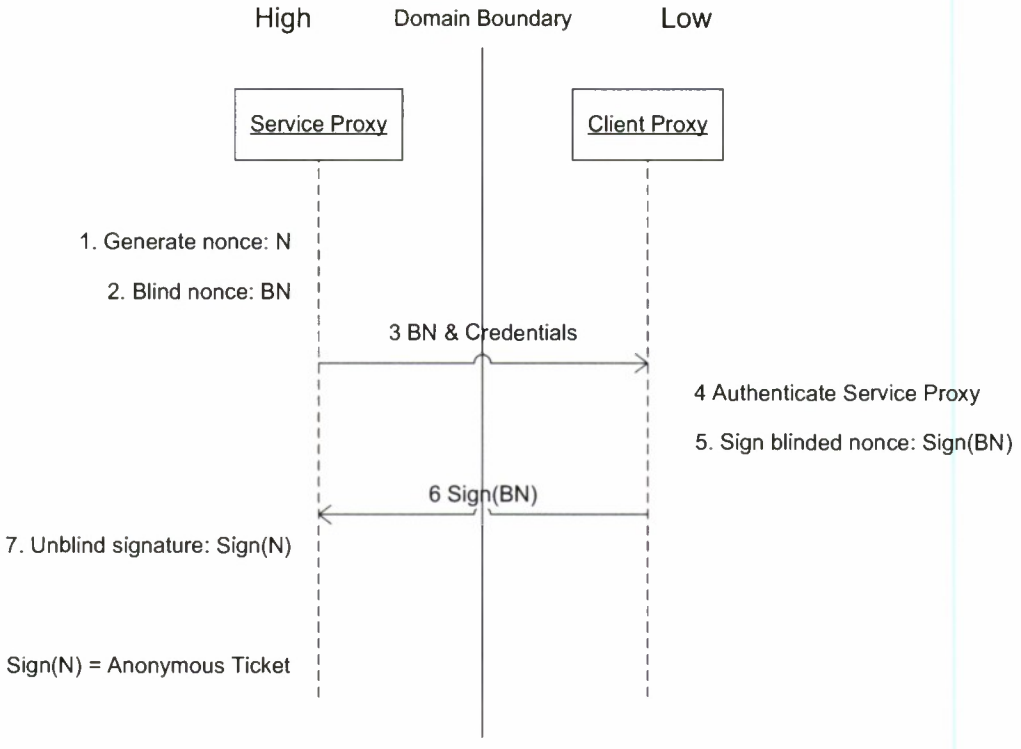

Figure 7: Anonymous ticket generation.

During service invocation, presentation of a nonce and its valid signature confirms that the service proxy has previously proved its identity. However, the client proxy cannot determine which service proxy is currently making the request out of all the service proxies that has previously obtained anonymous tickets. Using blind signatures, the client proxy does not actually see the content of the anonymous tickets it issues. It can only associate identifying information with the blinded nonce and the blinded signature. Without the secret blinding factor, it is cryptographically impossible to derive either the nonce or the signature of the nonce from their blinded counterparts. Therefore, anonymity does not depend on the trustworthiness or cooperation of the client proxy or any other components in Low.

The degree of anonymity depends on the total number of client High domains with tickets. High domains should obtain tickets in every cycle even if it does not utilize the service in order to help preserve anonymity of their peers. Tickets should be obtained en masse and stored for future service invocations. Requesting tickets in an as-needed basis will allow the Low domains to correlate ticket requests, which carry identity information, with service invocations.

Unlike normal signed messages, information cannot be carried in the nonce because the signer cannot examine its content. The only way to convey information in the anonymous ticket is through the fact that they are signed by a specific key. This affects the granularity of the trust relationship that could be described. It is impossible to sign nonces with an artificial caveat without a specific key pair for the caveat. For example, it would be impossible to say a domain can only access a subset service without a signature key pair specifically for that attribute. In the simplest case, a single key pair can grant access to the entire domain. This would be appropriate for domains providing non-sensitive services such as weather and mapping. Key pairs could also be established to grant access to individual service in the domain. In the most complex case, key pairs could be established for each possible attribute or sets of attributes of the client domain. These are attributes for the domain, such as the nationality or military branch of the domain. They do not refer to the security attributes of individual users such as rank or clearance level. Multiple tickets attesting to different attributes would need to be submitted for each service invocation to describe the trust relationship. This, of course, would reduce anonymity because

limited number of domains would have the same exact trust relationship. The tradeoff between granularity of the trust description and anonymity has to be decided by the service provider and the client domains when the federation is established.

Revocation of anonymous tickets can only be done in an all-or-none manner. It is impossible to only revoke anonymous tickets given to a single domain because by design, the client proxy cannot differentiate between anonymous tickets given to different domains. Therefore, revocation requires that all outstanding tickets must be invalidated by changing the signature keys. This would appear to be a large burden. However, revocation should happen very rarely in this scheme. Tickets authenticate service proxies that represent entire domains, not individual users. Revocation is only necessary if the MOU between federation partners changes, or if tickets are stolen.

Anonymous tickets should be used only one time. Otherwise, Low domain will be able to determine that multiple requests came from the same client domain. The burden is on the service proxy in the High domain to discard used tickets. This is indeed a security vulnerability at the infrastructure level. Unless the service proxy is implemented as a trusted component, there is no way to guarantee that tickets would not be reused, or that the nonces themselves do not contain messages. However, there are far easier ways of conveying the same amount of information at the service level. For example, clients can simply add a superfluous parameter to service invocations. If the parameter is an arbitrary number, the CDS would have no reason to consider it unreleasable. In order to send information through the anonymous tickets, attackers would have to completely compromise the service proxy, a component that is presumably tightly secured. This is indeed a potential for covert channels. However, it is a relatively minor threat in light of the large overt channels that exist in order to allow service invocations to cross domain boundaries.

In standard SOA, signatures provide authentication and protect the integrity of the message. Anonymous tickets provide authentication, but it must be incorporated into a special authentication block to provide message integrity. The client proxy needs to establish and maintain a public/private key pair specifically for the authentication block. The authentication block is created by 1) hashing the service request and client identity information, 2) combining that with one or more anonymous tickets, and then 3) encrypting it with the client proxy authentication block public key.

$$AB = Encrypt_{pub}\,(Hash(SR+ ID) + T)$$

Where: AB = authentication block
pub = client proxy AB public key
SR = service request
ID = identity information (sanitized)
T = anonymous ticket

The authentication block is sent along with the service request and identity information to the client proxy. Only the client proxy can decrypt it with its private key. The hash can be used to verify the integrity of the service request and sanitized identity information. The anonymous ticket will authenticate the service invocation.

If anonymous tickets are stolen, they can be used to gain unauthorized access to service domains. This carries a similar risk compared to the compromise of a private key, thus, the same safeguards should be in place. The anonymous ticket is always encrypted in the authentication block while in transit and is not vulnerable to man-in-the-middle attacks. Replay attacks can be prevented by including a timestamp in the service request and maintaining the list of used/invalided tickets in the client proxy.

The anonymous authentication algorithm ensures that the domain identity is not revealed by the protocol. However, components in Low domains can still determine the identity of High domain based on lower level network connection information. It is up to the CDS in the MLS infrastructure to hide the details of network connections.

5.3. Cross-Domain Service Discovery

Our preferred approach for cross-domain service discovery is the replication of the registry information from Low service repository to High repository. This eliminates the need for cross-domain interaction during the service discovery phase. In this scenario, High repositories will hold service descriptions for both High services and Low services. When High clients run a query, they will get matching services from both High and Low domains. The standard UDDI replication protocol can be used to replicate registry information from Low to High with minimal security risks. Data flow from Low to High is allowed under the standard Bell LaPadula model.

For scenarios where the replication is not feasible, service queries from High would have to be sent to repositories in Low. The UDDI interface needs be wrapped in a web service. This way, the protocol for anonymous service invocation can be applied. The Low repositories will not be able to tie queries to specific High clients or domains. Connection information for the UDDI interface web service can be distributed during ticket generation.

The MLS-SOA protocol requires that clients connect through the proxies when invoking services in a lower security level. After the discovery phase, the client should possess a modified connection string with appropriate endpoints and parameters to make the service invocation. A filter is needed to modify service descriptions. The location and implementation of the registry filter is flexible. It can filter incoming registry data from other domains before they are put into the registry, filter the outgoing query responses, moderate the query process as a component inside UDDI, or modify the requests coming out of the client.

5.4. Interoperation with Existing SOA

The components needed to support MLS-SOA are added cleanly on top of the existing MSL and SOA infrastructure. Interaction between the MLS-SOA components and the existing SOA components are done exclusively through the standard SOA/SAML interfaces. As far as the SOA components are concerned, the MLS-SOA components behave exactly like the regular SOA counterparts.

Installation and integration of the MLS-SOA into the SOA infrastructure involves the establishment of trust by exchanging certificates between interacting components. This is more or less identical to the steps involved in integrating an additional partner domain. Additional operations to support MLS happen behind the scene amongst the MLS-SOA components. Existing SOA components do not have to be modified. Also the existing single-level SOA applications would not be affected at all. Applications will be able to utilize services in the lower security domains transparently through the MLS-SOA infrastructure

6. Implementation

The MLS-SOA prototype was implemented in Java using Tomcat [19], OpenSAML [20], and BouncyCastle [21] libraries. Three separated networks were setup to simulate Top-Secret, Secret, and Unclassified domains. Several identity providers were deployed and tested [22-24]. MLS-SOA components were also setup around the public domain, namely the Internet. The service side MLS-SOA components do not necessarily have to be deployed by the owners of the services. In this case, they are deployed by the sensitive domains to securely access services available in the Internet. The address translation component was implemented as a network proxy for the browser. Since MLS-SOA operates at the HTTP/HTTPS level, it can be used to browse the Internet from systems in High. The address proxy transparently routs web browsers requests through the MLS-SOA infrastructure.

The rest of the section describes the MLS-SOA protocol for cross-domain service invocation in detail (Figure 8). The discovery phase is not shown. It assumes the client has gone through the discovery phase and has a valid connection string. It also assumes that the server proxy has gone through the ticket generation phase and is in possession of an anonymous ticket for authenticating with the client proxy.

Traffic that crosses the domain boundary is mediated by the MLS boundary controller. It is assumed that the boundary controller is configured for MLS-SOA traffic.

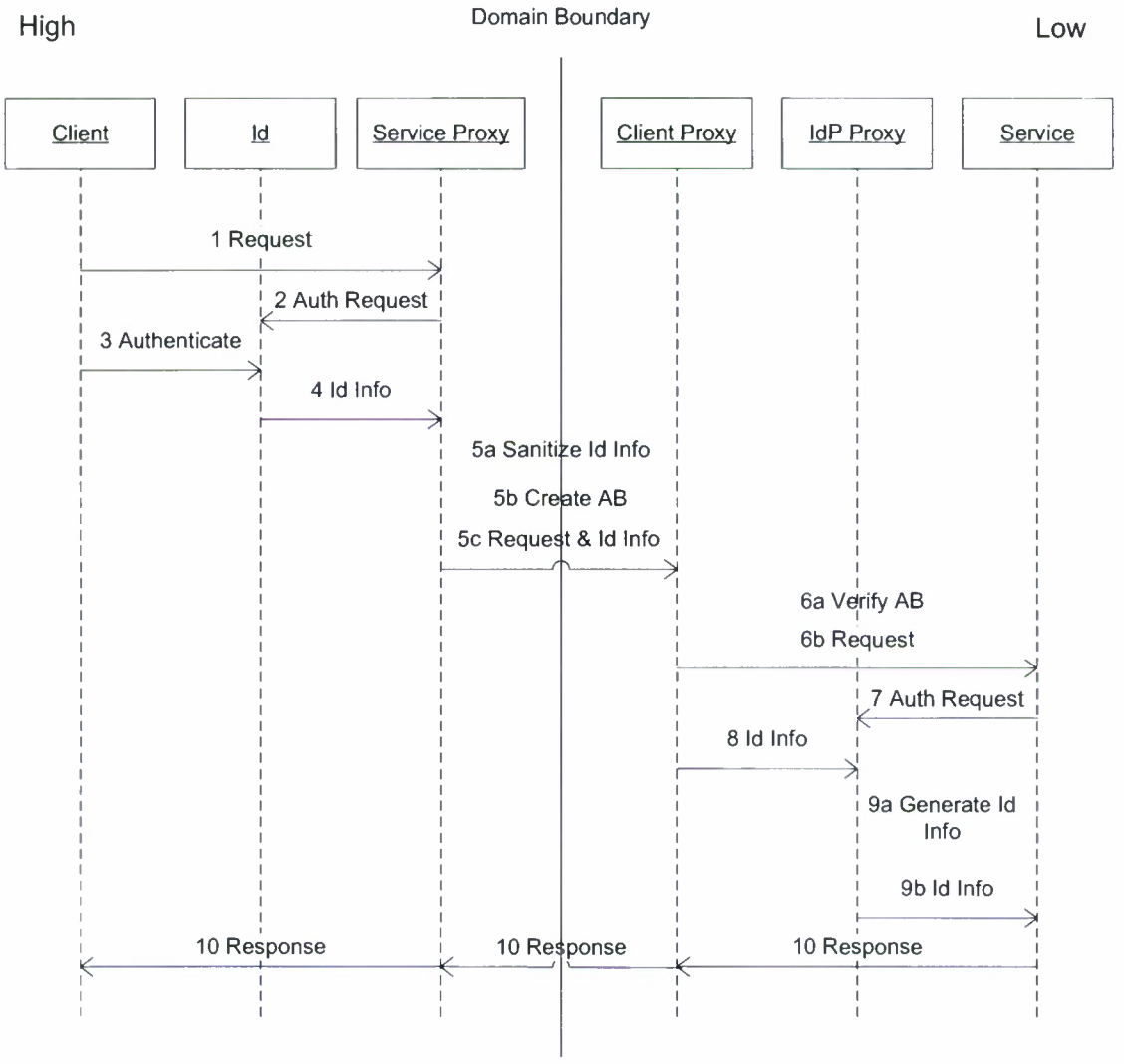

Figure 8: Detailed MLS-SOA service invocation protocol

1. The client sends service request to the service proxy.
2. The service proxy redirect client to the identity provider with a SAML authentication request.
3. The client authenticates with the identity provider.
4. The client is redirected back to the service proxy with identity information in a SAML authentication response.

5a. The service proxy sanitizes the identity information.

5b. The service proxy creates the authentication block.

   i. Create the hash of the service request and identity information.
   ii. Retrieve an anonymous ticket for the service provider domain from its ticket store.
   iii. Lookup the AB public key of the service provider domain client proxy.
   iv. Encrypt the hash and the anonymous ticket with the AB public key.

5c. The service request, identity information, and the authentication block are sent through the CDS to the client proxy in the service provider domain.
6a. The client proxy verifies the authentication block
   i. Decrypt the authentication block with its AB private key.
   ii. Authenticate the anonymous ticket by verifying the signature of the nonce.
   iii. Verify the integrity of the service request and identity information by checking the hash.
6b. The client proxy sends the service request to the service provider.
7. The service provider redirects the client proxy to the identity provider proxy with a SAML authentication request.
8. The identity provider proxy retrieves the client sanitized identity information from the client proxy.
9a. The identity provider proxy generates a new set of identity information for the service request.
9b. The client proxy is redirected back to the service provider with the generated identity information in a SAML authentication response.
10. The service provider authorizes the generated identity information and grants access. The service response is sent back to the client through the client proxy, the CDS, and the service proxy.

7. Discussion

The system described in this paper satisfies both the functional requirements for SOA and the security requirements of MLS. The core functionalities of the SOA paradigm are fully retained. Since requests and responses are supported at the HTTP/HTTPS level, MLS-SOA will also support regular web traffic. The proxy approach allows for the use of a customized interface across domain boundaries while also providing the standard interfaces expected by existing SOA components. Traffic that crosses the domain boundary is minimized to a single round trip per service invocation. In the standard SAML protocol, at least two round trips are required. We chose SAML as the standard for federated identity management in the prototype, but others such as OpenID [25], WS-Trust [26] or a customized protocol could also be used.

Anonymity provided by the blind signature protocol is cryptographically secure. It does not depend on the integrity or cooperation of any components in Low. Compromise of the client proxy in Low will not reveal non-releasable identity information associated with service invocations. Inference attacks based on attributing multiple service invocations to the same origin are thwarted by the strong anonymity. Compromise of the service proxy in High could result in the breach of MLS security. However, there is risk inherent in allowing service invocations from High to Low. MLS-SOA offers a practical solution that provides a balance between functionality and security.

Deployment of MLS-SOA infrastructure will not affect the existing single-level SOA operations. There is very little risk in deploying the MLS-SOA system for service providers in Low. Many more services from Low domains will be made available to clients in High. Clients in High must make a tradeoff between functionality and security. The existing CDS will govern the release of information through both overt and covert channels. The MLS-SOA security architecture will help to mitigate covert channels, prevent the release of identity information, and thwart inference attacks. However, the threats cannot be completely eliminated. In most cases, the gain in functionality should outweigh the security risk.

Capabilities developed for MLS-SOA can be useful for other scenarios with similar requirements. The proxy architecture decouples the interfaces used for intra-domain and cross-domain interaction. This allows for simplification of communications across firewalls and VPN's while maintaining compatibility with existing interfaces. The anonymous authentication scheme can be useful for scenarios such as pervasive computing that require both strong authentication and anonymity.

8. Related Work

There is a great deal of interest in deploying SOA in military information systems. A conceptual design was presented for full MLS support at the service level of SOA [27]. Their approach uses security meta-data labeling and mandatory access control to enforce MLS policy. This approach calls for the creation of an entirely new trusted computing platform for SOA requiring a large number of specialized high assurance devices. The development and accreditation cost of such a system would be astronomical. Our approach of using an MSL implementation supports MLS capabilities leverages existing technology and does not require any additional high-assurance components. It is much more practical in terms of cost, ease of integration, and risk to existing operations.

The MLS problem has also been examined in the related field of distributed computing [28, 29]. However, the security issues central to SOA, namely covert channels, release of identity information and inference attacks, have not been directly explored.

The inference and aggregation problem has been studied a great deal for many decades in the context of multi-level secure database management [13, 14]. However, they apply to the context of relational databases and do not readily translate to the SOA environment.

9. Conclusion

The MLS-SOA architecture presented in this paper offers a practical solution for supporting SOA in an MLS environment. It maintains full compatibility with existing SOA infrastructure and technology. Standard CDS that are already part of the MLS infrastructure can be used. MLS-SOA components reside completely inside single security levels and do not require high-assurance. The existing CDS only needs to be reconfigured to process MLS-SOA traffic across domain boundaries. Security properties of the system do not depend on the integrity or the cooperation of components in Low. The anonymity and non-inference properties of the system are cryptographically secure.

The MLS-SOA approach is preferable to the alternatives, 1) the status quo of single-level SOA, and 2) development of a complete TCP for the SOA infrastructure. The MLS-SOA design can be productized and deployed to military networks in a very short time. The system is very simple and straightforward to install and operate. It can be added on top of existing SOA and MLS infrastructure without impacting existing operations. It can be deployed by service providers and clients independently to add MLS capabilities to their respective domains with minimal disruptions.

10. References

1. Birman, K., R. Hillman, and S. Pleisch, *Building Network-Centric Military Applications over Service Oriented Architectures.* Proceedings of the SPIE Conference on Defense Transformation and Network-Centric Systems, 2005.
2. Bell, D.E. and L.J. LaPadula, *Secure computer systems: Secure Computer System: Unified Exposition and Multics Interpretation.* 1976, The Mitre Corp.
3. Goguen, J.A. and J. Meseguer, *Security Policies and Security Models.* Security and Privacy, IEEE Symposium on Security and Privacy, 1982.
4. Saydjari, O.S., *Multilevel Security: Reprise.* IEEE Security and Privacy, 2004. **2**(5): p. 64-67.
5. Rushby, J., *Design and Verification of Secure Systems.* Proceedings of 8th ACM Symposium on Operating System Principles, 1981.
6. Harrison, W.S., et al., *The MILS Architecture for a Secure Global Information Grid.* CrossTalk, 2005. **18**(10): p. 20-24.
7. OASIS, *UDDI Version 3.0.2.* 2004.
8. OASIS, *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0.* 2005.
9. Gudgin, M., et al., *SOAP Version 1.2.* W3C, 2007.

10. DISA IASE. *DISA CDS.* [cited 2009 4/25]; Available from: http://iase.disa.mil/cds/index.html.
11. Goertzel, K.M., *Cross-Domain Controlled Interface and Labeling (CDCIL) Services*. 2005, Booz Allen Hamilton.
12. Kang, M.H., I.S. Moskowitz, and D.C. Lee, *A Network Pump.* IEEE Transactions on Software Engineering, 1996. **22**(5).
13. Marks, D.G., *Inference in MLS database systems.* IEEE TDKE, 1996. **8**(1).
14. Garvey, T.D., *The inference Problem for Computer Security.* 1992, SRI International.
15. Chaum, D., *Blind Signatures for Untraceable Payments.* Advances in Cryptology, 1982.
16. Ren, K., et al., *A Novel Privacy Preserving Authentication and Access Control Scheme for Pervasive Computing Environments.* IEEE Transactions on Vehicular Technology, 2006. **55**(4).
17. Chaum, D., *Security without Identification: Transaction Systems to make Big Brother Obsolete.* Communications of the ACM 1985. **28**(10).
18. Chaum, D., *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms.* Communications of the ACM, 1981. **24**(2).
19. The Apache Software Foundation. *Apache Tomcat.* [cited 2009 2/29]; Available from: http://tomcat.apache.org/.
20. Internet2. *OpenSAML.* [cited 2009 2/25]; Available from: https://spaces.internet2.edu/display/OpenSAML/Home/.
21. The Legion. *Bouncy Castle.* [cited 2009 4/25]; Available from: http://www.bouncycastle.org/.
22. Kim, A., A. Khashnobish, and M.H. Kang, *An Architecture for Web Services Authentication and Authorization in a Maritime Environment.* ITNG, 2007.
23. *Shibboleth.* [cited 2009 5/24]; Available from: http://shibboleth.internet2.edu/.
24. *OpenSSO.* [cited 2009 5/24]; Available from: https://opensso.dev.java.net/.
25. OpenID Foundation, *OpenID.* 2007.
26. OASIS, *WS-Trust 1.3.* 2007.
27. Ramasamy, H.V. and M. Schunter, *Multi-Level Security for Service-Oriented Architectures.* Military Communications Conference MILCOM, 2006.
28. Kang, M.H., et al., *A Multilevel Secure Workflow Management System.* CAiSE, 1999.
29. Kang, M.H., J.N. Froscher, and B.J. Eppinger, *Towards an infrastructure for MLS distributed computing.* Proceedings of the 14th Annual Computer Security Applications Conference, 1998.